局部线性嵌入降维算法

Locally linear embedding Algorithm

Arrow Luo

2016 年 10 月 19 日

1 算法介绍

局部线性嵌入算法（Locally linear embedding, LLE）是一个非线性降维方法，由 Sam T.Roweis 和 Lawrence K.Saul 于 2000 年提出并发表在《Science》杂志上[1]。它能够使降维后的数据保持原有拓扑结构不变。现在已经广泛应用于图像数据的分类与聚类、文字识别、多维数据的可视化、以及生物信息学等领域中。

LLE 算法可以可以用图1 来描述。在图1 中，LLE 能成功地将三维非线性数据（中间散点图数据）映射到二维空间（最后的平面散点图）中。如果将图1 中红颜色和蓝颜色的数据分别看成是分布在三维空间中的两类数据，通过 LLE 算法降维后，数据在二维空间中仍能保持相对独立的两类。

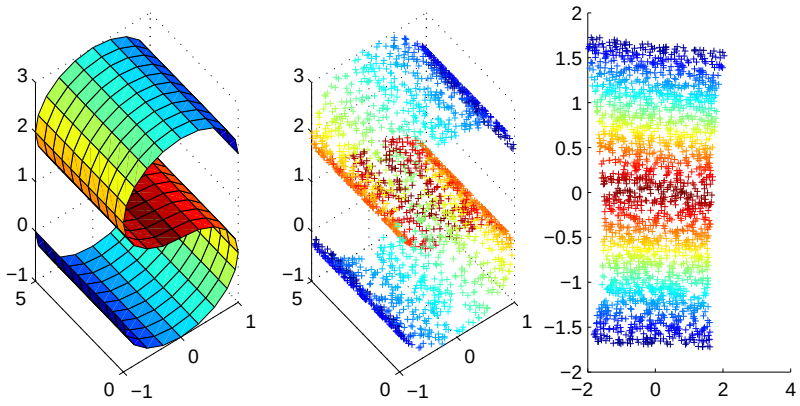

图 1. LLE 非线性降维示例

LLE 是一种局部算法。它的主要思想是利用数据的局部线性来逼近全局线性：即假设任意样本点都可表示为其临近样本点的线性组合，在寻找数据的低维嵌入同时，保持这种邻域线性组合关系不变。

具体来说，算法假定平滑流形在局部具有线性性质，一个点可以通过其局部邻域内的点重构出来。首先它会将式（1）最小化。

$$\varepsilon(W) = \sum_i |\vec{X}_i - \sum\nolimits_j W_{ij}\vec{X}_j|^2 \tag{1}$$

以求解出最优的局部线性重构矩阵 W ，对于距离较远的点 i 和 j，W_{ij} 应当等于零。这之后再把 W

当作已知量对式（2）进行最小化。

$$\Phi(Y) = \sum_i |\vec{Y_i} - \sum\nolimits_j W_{ij}\vec{Y_j}|^2 \tag{2}$$

这里 Y 成了变量，亦即降维后的向量，对这个式子求最小化的意义很明显，就是要求如果原来的数据 X_i 可以以 W 矩阵里对应的系数根据其邻域内的点重构出来的话，那么降维过后的数据也应该保持这个性质 (可以在 http://blog.pluskid.org/?p=290 了解更多)。

2 算法步骤

LLE 算法可以归结为三步:

1) 寻找每个样本点的 k 个近邻点；
2) 由每个样本点的近邻点计算出该样本点的局部重建权值矩阵；
3) 由该样本点的局部重建权值矩阵和其近邻点计算出该样本点的输出值；

具体的算法流程如图2 所示[1]。

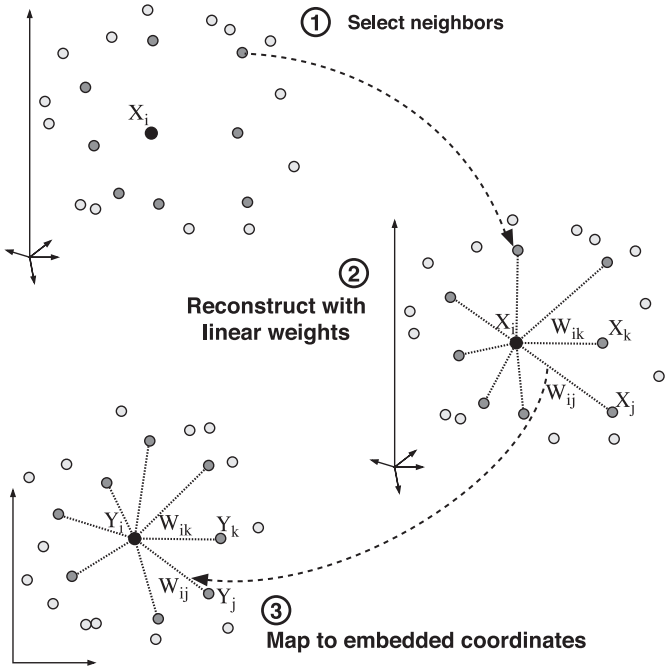

图 2. LLE 算法实现步骤

3 LLE 伪代码

算法 1 LLE Algorithm

1: **Input:** **X** D by N matrix consisting of N data items in D dimensions; **K** the number of neighbours in first step.

2: **Output:** **Y** d by N matrix consisting of $d < D$ dimensional embedding coordinates for the input points.

3: **Find neighbours in X space.**

- for $i = 1 : N$
- compute the distance from X_i to every other point X_j
- find the K smallest distances
- assign the corresponding points to be neighbours of X_i
- end

4: **Solve for reconstruction weights W.**

- for $i = 1 : N$
- create matrix Z consisting of all neighbours of X_i
- subtract X_i from every column of Z
- compute the local covariance $C = Z' * Z$
- solve linear system $C * w = 1$ for w
- set $W_{ij} = 0$ if j is not a neighbor of i
- set the remaining elements in the i_{th} row of W equal to $w/sum(w)$;
- end

5: **Compute embedding coordinates Y using weights W.**

- create sparse matrix $M = (I - W)' * (I - W)$
- find bottom $d + 1$ eigenvectors of M (corresponding to the $d + 1$ smallest eigenvalues)
- set the q_{th} ROW of Y to be the $q+1$ smallest eigenvector (discard the bottom eigenvector $[1, 1, 1, 1...]$ with eigenvalue zero)

该算法的 matlab 代码可以到https://github.com/ArrowLuo/LLE_Algorithm下载，也可以直接到http://www.cs.nyu.edu/~roweis/lle/code.html下载。

4 LLE 算法原理[2]

4.1 LLE 重构矩阵计算

上面提到 LLE 是一种局部算法，它主要利用数据的局部线性来逼近全局线性。所以 LLE 的第一步就是寻找每个样本点的 K 个近邻点；直观地，对于每个样本点 X_i，根据参数 K 计算出该样本点的领域点集 $\{Z_{i1}, ..., Z_{iK}\}$

接下来就是计算出每个样本点的局部线性重构矩阵 W_{ij}；目标函数如公式（3）所示。

$$\begin{aligned} &\arg\min_{W} \sum_{i=1}^{N} \left\| X_i - \sum_{j=1}^{K} W_{ij} Z_{ij} \right\|^2 \\ &s.t. \sum_{j=1}^{K} W_{ij} = 1, i = 1, ..., N \end{aligned} \tag{3}$$

N 为样本系数，W_{ij} 为样本点 i 对应的第 j 个重构系数。
观察单个样本点的目标函数。

$$\begin{aligned} \varepsilon &= \left\| X_i - \sum_{j=1}^{K} W_{ij} Z_{ij} \right\|^2 = \left\| \sum_{j=1}^{K} (W_{ij} X_i - W_{ij} Z_{ij}) \right\|^2 \\ &= \left\| \sum_{j=1}^{K} W_{ij} (X_i - Z_{ij}) \right\|^2 = \|Q_i W_i\|^2 \end{aligned} \tag{4}$$

式子中 $Q_i = [X_i - Z_{i1}, X_i - Z_{i2}, \cdots, X_i - Z_{iK}]$，重构系数向量 $W_i = [W_{i1}, W_{i2}, \cdots, W_{iK}]$。为使 ε 最小，可以用拉格朗日乘数法求解 W_i 的值。

$$\begin{aligned} \mathcal{L}_i &= \left\| X_i - \sum_{j=1}^{K} W_{ij} Z_{ij} \right\|^2 - \lambda_i (\sum_{j}^{K} W_{ij} - 1) \\ &= \|Q_i W_i\|^2 - \lambda_i 1^T W_i \\ &= W_i^T Q_i^T Q_i W_i - \lambda_i 1^T W_i \end{aligned} \tag{5}$$

其中 $1 = [1, 1, \cdots, 1]$，长度为 K。
将 $\mathcal{L}_i$ 对 W_i 求偏导。

$$\frac{\partial \mathcal{L}_i}{\partial W_i} = Q_i^T Q_i W_i - \lambda_i 1 = C_i W_i - \lambda_i 1 = 0 \tag{6}$$

注：矩阵偏导请参考 *[2]* 附录，此处不详述。
推出

$$W_i = \lambda_i C_i^{-1} 1$$

式子中 $C_i = Q_i^T Q_i$；另外有 $\sum_{j=1}^{K} W_{ij} = 1^T W_i = 1$，则

$$1^T \lambda_i C_i^{-1} 1 = 1 \Rightarrow \lambda_i = (1^T C_i^{-1} 1)^{-1}$$

最终

$$\begin{aligned} W_i &= \lambda_i C_i^{-1} 1 = (1^T C_i^{-1} 1)^{-1} C_i^{-1} 1 \\ &= \frac{C_i^{-1} 1}{1^T C_i^{-1} 1} \end{aligned} \tag{7}$$

上式中 $K > D$ 时 C_i 不一定可逆，可以通过加入 C 的迹的小倍数来规范化。

$$C_i = C_i + rI$$

这就是下面 matlab 代码的来由。

```
if(K>D)
  fprintf(1,'   [note: K>D; regularization will be used]\n');
  tol=1e-3; % regularlizer in case constrained fits are ill conditioned
else
  tol=0;
end
W = zeros(K,N);
for ii=1:N
   z = X(:,neighborhood(:,ii))-repmat(X(:,ii),1,K); % shift ith pt to origin
   C = z'*z;                                        % local covariance
   C = C + eye(K,K)*tol*trace(C);                   % regularlization (K>D)
   W(:,ii) = C\ones(K,1);                           % solve Cw=1
   W(:,ii) = W(:,ii)/sum(W(:,ii));                  % enforce sum(w)=1
end;
```

4.2 LLE 最优嵌入计算

计算出重构矩阵 W 之后，由于 W 中只是包含了相邻点间的权重系数，将不相邻的权重系数置为 0，然后由该重建权值矩阵和其近邻点计算出样本点的输出值；就是求解最优 Y 值使下面 $\Phi(Y)$ 最小。

$$\begin{aligned} &\arg\min_{Y} \sum_{i} \left\| Y_i - \sum_j W_{ij} Y_j \right\|^2 \\ &s.t. \sum_{i=1}^{N} Y_i = 0, \\ &N^{-1} \sum_{i=1}^{N} Y_i Y_i^T = I_d. \end{aligned} \tag{8}$$

式子中 $Y = [Y_1, Y_2, \cdots, Y_N]$ 是 $d \times N$ 矩阵。d 是 Y 的维数。

首先解释加入约束条件

$$\begin{cases} \sum_{i=1}^{N} Y_i = 0 \\ N^{-1} \sum_{i=1}^{N} Y_i Y_i^T = I_d \end{cases}$$

的原因。

根据文章 [3] 的解释，加入这两个约束条件是在不影响损失函数求解的前提下使方程易于求解（This optimization is performed subject to constraints that make the problem well posed.）。根据文章 [2] 的解释，加入约束条件的目的是保证解得唯一性（guarantee the essential uniqueness of the solution）。

实质上第一个约束是所有点的均值。第二个约束是解的协方差。所有点的均值为 0，协方差为单位阵 I。这就使得解空间分布唯一。

将目标函数转换为矩阵表示

$$\begin{aligned} &\arg\min_{Y} \|Y - YW^T\|^2 \\ &s.t. Y 1_N = 0, \\ &N^{-1} Y Y^T = I_d. \end{aligned} \tag{9}$$

同样应用拉格朗日数乘法对上述最优化问题求解。

$$\mathcal{L} = \sum_{i} \left\| Y_i - \sum_j W_{ij} Y_j \right\|^2 - \sum_{\alpha,\beta=1}^{d} \lambda_{\alpha\beta} [\frac{1}{N} \sum_{i=1}^{N} y_{\alpha i} y_{\beta i} - \delta_{\alpha\beta}] - \sum_{\alpha=1}^{d} \kappa_\alpha \sum_{i=1}^{N} y_{\alpha i} \tag{10}$$

其中，$\lambda_{\alpha\beta}$、κ_α 是拉格朗日乘子；并且 $\lambda_{\alpha\beta}=\lambda_{\beta\alpha}$，如果 $\alpha=\beta$ 时 $\delta_{\alpha\beta}=1$，否则 $\delta_{\alpha\beta}=0$。
为了将上式用矩阵表示，使 $\Lambda=[\lambda_{\alpha\beta}]_{1\le\alpha,\beta\le d}\in\mathbb{R}^{d\times d}$ 且 $\kappa=[\kappa_1,\kappa_2,\cdots,\kappa_d]$。

$$\mathcal{L}=\|Y-YW^T\|^2-\Lambda[\frac{1}{N}YY^T-I_d]-1_N^TY\kappa$$

对 Y 求偏导并令为 0，得到

$$\frac{\partial\mathcal{L}}{\partial Y}=Y(I_N-W^T)(I_N-W)-N^{-1}\Lambda Y-\kappa 1_N^T=0 \tag{11}$$

注：矩阵偏导请参考 *[2]* 附录，此处不详述。
由于 $(I_N-W)1_N=0$ 和 $Y1_N=0$，得到 $\kappa 1_N^T=0$，故 $\kappa=0$，将 $\kappa=0$ 带入（11）得到

$$Y(I_N-W^T)(I_N-W)=N^{-1}\Lambda Y$$

将上式左右转置，

$$(I_N-W^T)(I_N-W)Y^T=N^{-1}Y^T\Lambda$$

上式中 Λ 和 Y 均为未知变量，但是 Λ 是对称阵，一定能够找到正交阵 U 使得 Λ 对角化 $\Lambda=UDU^T$。同时令 $\hat{Y}=U^TY$，则

$$(I_N-W^T)(I_N-W)(Y^TU)=N^{-1}(Y^TU)D\iff(I_N-W^T)(I_N-W)\hat{Y}^T=N^{-1}\hat{Y}^TD$$

值得一提的是，当令 $\hat{Y}=U^TY$ 时，

$$\begin{aligned}&\arg\min_{\hat{Y}}\|\hat{Y}-\hat{Y}W^T\|^2\\&s.t.\hat{Y}1_N=0,\\&N^{-1}\hat{Y}\hat{Y}^T=I_d.\end{aligned} \tag{12}$$

依然成立。
因为 $N^{-1}\hat{Y}\hat{Y}^T=I_d$，故组成 $\hat{Y}$ 矩阵的列向量模长为 $N^{1/2}$。所以 $N^{-1/2}\hat{Y}$ 才是 $(I_N-W^T)(I_N-W)$ 的特征向量。
到这里就可以得到求解 $\Phi(Y)$ 最小值对应的 Y 为：$(I_N-W^T)(I_N-W)$ 的最小 d 个特征值对应特征向量再乘上常数 $N^{1/2}$。
事实上，因为 $(I_N-W^T)(I_N-W)$ 是半正定矩阵，它的特征值 λ_i 一定非负，由 $W1_N=I_N$ 可知，$(I_N-W^T)(I_N-W)$ 的最小特征值为 0，对应的特征向量为 1_N。按照上面陈述，$N^{1/2}1_N$ 是 Y 的一个向量。但是它不满足 $\hat{Y}1_N=0$，所以应该舍去最小的特征值。
所以，求解 $\Phi(Y)$ 最小值对应的 Y 为：$(I_N-W^T)(I_N-W)$ 的最小 $d+1$ 个特征值对应特征向量再乘上常数 $N^{1/2}$，舍掉最小的特征值对应的特征向量。
另一个方面，由矩阵范数与迹的关系 $\|A\|_F=\sqrt{tr(A^TA)}=\sqrt{tr(AA^T)}$

$$\begin{aligned}\sum_i\big\|Y_i-\sum\nolimits_j W_{ij}Y_j\big\|^2&=\|Y-YW^T\|^2=tr\big((Y-YW^T)(Y-YW^T)^T\big)\\&=tr\big(Y(I-W^T)(I-W)Y^T\big)\end{aligned} \tag{13}$$

从这个公式可知 $\Phi(Y)$ 最小值为 $(I-W^T)(I-W)$ 对应的特征值之和，由于 $(I-W^T)(I-W)$ 的最小特征值为 0，所以前 $d+1$ 个特征值之和与除去 0 特征值的前 d 个特征值之和相同。而且 0 特征值对应的特征向量为常数向量 *1*。与上面表述印证。
这就是下面 matlab 代码的来由。

```
M = sparse(1:N,1:N,ones(1,N),N,N,4*K*N);
for ii=1:N
   w = W(:,ii);
   jj = neighborhood(:,ii);
   M(ii,jj) = M(ii,jj) - w';
   M(jj,ii) = M(jj,ii) - w;
   M(jj,jj) = M(jj,jj) + w*w';
end;

% CALCULATION OF EMBEDDING
options.disp = 0;
options.isreal = 1;
options.issym = 1;
[Y,eigenvals] = eigs(M,d+1,0,options);
Y = Y(:,2:d+1)'*sqrt(N); % bottom evect is [1,1,1,1...] with eval 0
```

值得注意的是上述特征值的求解函数 eigs，因为 matlab 版本的问题导致 $2:d+1$ 不一定是除 0 外的 d 个特征向量，运行代码时需要打印查看特征值是否符合算法要求。

参考文献

[1] Roweis, S. T., & Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. Science, 290(5500), 2323-6.

[2] WOJCIECHCHOJNACKI, & Brooks, M. J. (2009). A note on the locally linear embedding algorithm. International Journal of Pattern Recognition & Artificial Intelligence, 23(8), 1739-1752.

[3] Saul, L. K., & Roweis, S. T. (2001). An introduction to locally linear embedding. Journal of Machine Learning Research, 7.